Finding a Curve in a Map*

Carola Wenk† Helmut Alt‡ Alon Efrat†
Lingeshwaran Palaniappan† Günter Rote‡

ABSTRACT

Given a polygonal curve and a geometric graph, we describe an efficient algorithm to find a path in the graph which is most similar to the curve, using the well-known Fréchet distance for curves.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Geometrical problems and computations*

General Terms

Algorithms

Keywords

Fréchet distance, shape matching

1. INTRODUCTION

Patterns consisting of line segments occur in many applications of a geometric nature, like computer vision, geographic information systems, CAGD, etc. In many cases the problem occurs to determine whether some given pattern H is equal to or similar to some part of a larger pattern G. Here, we define a feasible distance measure for the case that H is a polygonal curve and G is a geometric graph. This distance measure is a generalization of the well-known Fréchet distance for curves, which has been investigated before by Alt and Godau [2].

*This work is based on the algorithm published in [1].

†University of Arizona, Computer Science Department, 1040 E 4th Street, Tucson, AZ 85721, USA. {alon, carolaw, lingesh}@cs.arizona.edu

‡Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany. {alt,rote}@inf.fu-berlin.de

SoCG'03, June 8–10, 2003, San Diego, California, USA.
ACM 1-58113-663-3/03/0006.

DEFINITION 1 (FRÉCHET DISTANCE). *Let $f\colon I=[l_I,r_I]\to\mathbb{R}^2$, $g\colon J=[l_J,r_J]\to\mathbb{R}^2$ be two planar curves, and let $\|\cdot\|$ denote the Euclidean norm. Then the* Fréchet distance $\delta_F(f,g)$ *is defined as*

$$\delta_F(f,g) := \inf_{\substack{\alpha\colon [0,1]\to I\\ \beta\colon [0,1]\to J}} \max_{t\in[0,1]} \|(f(\alpha(t)), g(\beta(t))\|.$$

where α and β range over continuous and non-decreasing reparameterizations with $\alpha(0)=l_I$, $\alpha(1)=r_I$, $\beta(0)=l_J$, $\beta(1)=r_J$.

We consider a given polygonal curve $\alpha\colon[0,p]\to\mathbb{R}^2$, and a geometric graph $G=(V,E)$ embedded in the plane with non-crossing straight-line edges. We wish to find a path π in G such that the Fréchet distance $\delta_F(\alpha,\pi)$ is minimized. This is a partial matching variant. This definition allows a path π in G to travel the same edge in G several times. It may even directly return on the same edge, but only after visiting a vertex. We assume that π starts and ends in a vertex, although this constraint can be easily relaxed.

In [1] we presented an algorithm which computes the optimal distance in $O(mn\log(mn)\log n)$ time if α consists of m segments and G has n edges. For a given $\varepsilon>0$, the *decision problem* of deciding whether there exists a path π in G such that $\delta_F(\alpha,\pi)\le\varepsilon$, can be solved in $O(mn\log n)$ time. This algorithm has been implemented in C with a graphical user interface using OpenGL. The program allows to edit the graph and the curve, to solve the decision problem, to perform binary search on δ_F, and it visualizes the computed feasible parameterizations in a walk-through animation. Even without any specific optimizations it runs surprisingly fast. In the accompanying video to this paper we illustrate the problem, sketch the algorithm, and show the implementation.

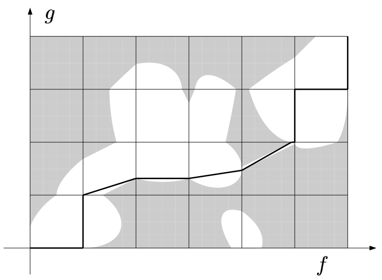
The problem in this form already has many applications. The following one, for example, looked particularly appealing to us: Consider a given roadmap, and a person travelling on some of the roads, while recording its position using a GPS receiver. The roadmap can be modelled by a planar embedded graph, and the path the person travelled is represented by a sequence of *GPS positions* recorded by the GPS receiver, which we connect by straight line segments to form a polygonal curve. Since the GPS receiver usually introduces noise the captured curve will not exactly lie on the roadmap. The task is to identify those roads which have actually been travelled. This is a prerequisite for incrementally constructing roadmaps from such GPS curves, which is especially interesting for roads such as hiking trails in a forest which are not visible on aerial pictures.

2. ALGORITHM

We first solve the decision problem. Afterwards we apply parametric search, similar as in [2], to eventually solve the minimization problem, but this extension has not been implemented.

2.1 Free Space Diagram

We employ the notion of the *free space* F_ε and the free space diagram FD_ε of two curves, which was introduced in [2]: Let $f: I \to \mathbb{R}^2$, $g: J \to \mathbb{R}^2$ be two curves; $I, J \subseteq \mathbb{R}$. The set $F_\varepsilon(f,g) := \{(s,t) \in I \times J \mid \|f(s) - g(t)\| \leq \varepsilon\}$ denotes the *free space* of f and g. We call the partition of $I \times J$ into regions belonging or not belonging to $F_\varepsilon(f,g)$ the *free space diagram* $FD_\varepsilon(f,g)$. We call points in F_ε *white* or *feasible* and points in $FD_\varepsilon \setminus F_\varepsilon$ *black* or *infeasible*. See the figure below for an illustration.



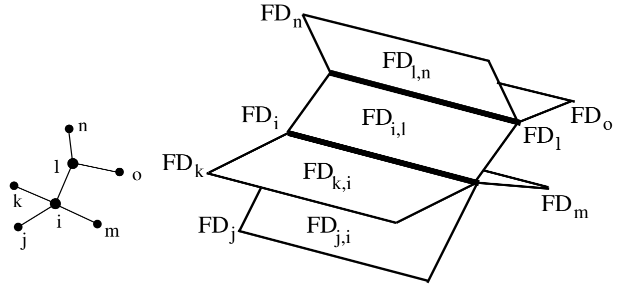
In [2] it has been shown that $\delta_F(f,g) \leq \varepsilon$ if and only if there exists a curve within $F_\varepsilon(f,g)$ from $(0,0)$ to $(1,1)$ which is monotone in both coordinates. We call a curve within $F_\varepsilon(f,g)$ *feasible*. Observe that the monotone curve in $F_\varepsilon(f,g)$ from $(0,0)$ to $(1,1)$ as a continuous mapping from $[0,1]$ to $[0,1]^2$ directly gives continuous increasing reparameterizations α and β.

2.2 Free Space Surface

For each edge $(i,j) \in E$ let $s_{i,j}: [0,1] \to \mathbb{R}^2$ be its embedding as an oriented straight-line segment. For every edge $(i,j) \in E$ consider the free space $F_{i,j} := F_\varepsilon(\alpha, s_{i,j}) \subseteq [0,p] \times [0,1]$. The free space diagram $FD_{i,j} := FD_\varepsilon(\alpha, s_{i,j})$ is the subdivision of $[0,p] \times [0,1]$ into the *white* points of $F_{i,j}$ and into the *black* points of $[0,p] \times [0,1] \setminus F_{i,j}$.

As shown in [2], $FD_{i,j}$ consists of a row of p cells. Each such cell corresponds to a line segment of α, and the free space in each cell is the intersection of an ellipse with that cell. $FD_{i,j}$ is the same as $FD_{j,i}$, except that the bottom and the top edge change roles. For a vertex $j \in V$ let $FD_j := FD_\varepsilon(\alpha, j)$, which is a one-dimensional free space diagram consisting of black or white intervals. Let $F_j := F_\varepsilon(\alpha, j)$ be the corresponding one-dimensional free space, which consists of a collection of white intervals. Furthermore, let $\mathbf{L}_j$ be the left endpoint and $\mathbf{R}_j$ be the right endpoint of FD_j.

For each vertex $i \in V$, the free space diagrams $FD_{i,j}$ for all adjacent vertices j have the one-dimensional free space diagram FD_i in common. Thus we can glue together the two-dimensional free space diagrams at the one-dimensional free space diagrams which they share, according to the adjacency of G. In this way we obtain a topological structure which we call the *free space surface* of G and α. See the next figure for an example.



2.3 Dynamic Programming and Sweep

The algorithm in [2] computes a monotone feasible path in the free space diagram of two polygonal curves in a dynamic programming fashion. We apply a related approach to our more general setting: We search for a feasible monotone path in the free space surface. This path has to start at some white left corner $\mathbf{L}_k$ and has to end at some white right corner $\mathbf{R}_j$, for two vertices $j, k \in V$, since the corresponding path π in G has to start and end in a vertex of G. Any path π in G selects a sequence of free space diagrams in the free space surface, whose concatenation yields $FD_\varepsilon(\alpha, \pi)$.

For every vertex $j \in V$ let $\mathcal{R}(j)$ be the set of all points $u \in F_j$ for which there exists a $k \in V$ and a path π from k to j in G such that there is a monotone feasible path from $\mathbf{L}_k$ to u in $F_\varepsilon(\alpha, \pi)$. We call an interval of points in $\mathcal{R}(j)$ *reachable*. We thus know that there is a path π in G with $\delta_F(\alpha, \pi) \leq \varepsilon$ iff there is a vertex $j \in V$ such that $\mathbf{R}_j \in \mathcal{R}(j)$.

Similar to [2] we first decide whether there is a feasible path in the free space surface by computing $\mathcal{R}(j)$ for all $j \in V$ in a dynamic programming manner. In fact we will not store the whole $\mathcal{R}(j)$ but only parts of it which allow us to arrive at the correct decision. The algorithm solving the decision problem consists of three stages: In the *preprocessing stage* we compute the free space diagrams $FD_{i,j}$ together with a data structure that supports reachability queries on the $FD_{i,j}$.

In the *dynamic programming stage* we decide whether there is a feasible monotone path in the free space surface. Conceptually we sweep all $FD_{i,j}$ at once with a vertical sweep line from left to right. For each $i \in V$ we store a set $C_i \subseteq \mathcal{R}(i)$ of white points which lie to the right of the sweepline, and for which the last segment of their associated feasible monotone path crosses or ends at the sweep line. We compute the C_i in a dynamic programming manner. We maintain a priority queue Q of white intervals of FD_i which are known to be reachable. Their left endpoints are the sweep events. During the sweep we update the C_i Dijkstra-style by querying the precomputed reachability data structures.

In the *path reconstruction stage*, if there is a $j \in V$ such that $\mathbf{R}_j \in C_j$, we reconstruct a path π in G along with feasible reparameterizations of π and α that witness the fact that $\delta_F(\alpha, \pi) \leq \varepsilon$.

3. REFERENCES

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. In *14th ACM-SIAM Sympos. Discrete Algorithms*, 2003, pp. 589–598.

[2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.* **5** (1995), 75–91.